Projects Summer Term 2025

Supervisors: Alexander Dinges, Cass Alexandru, Michael Youssef, Sebastian Schloßer

Kick-off: 30.04.2025

Organization

Schedule

- Kick-off: Wednesday, 30.04.2025, 13:45 - 15:15, room 36-265
- Final presentations: presumably Friday, 25.07.2025, time and room TBA
  Mandatory attendance!
- Regular meetings with your supervisor

Project Teams

- Four topics to choose from
- One or two teams per topic (working independently)
- Four or five students per team (collaborating)

Agenda

- We present you the topics
- You (physically) go to the supervisor whose topic you would like to work on
- There you form teams and exchange details with team members & supervisor

Data Management for ExClaim

Supervisor: Sebastian Schloßer

About ExClaim

- Exercise management
  - Students
  - Groups (same exercise session)
  - Teams (collaborate on homework)
  - Homework submission
  - Homework grading
  - Automatic group assignment
- Exam management
- Automatic test execution

Remote Test Executor (RTE)

- Docker
- Consistent environment
- No risks for tutors
- Fast feedback (for students and as guidance while grading)
- Can execute test cases without providing the source code to students

History (1)

- Softech Achievement Tracking System (STATS)
    - Manage students, groups, teams, exams
    - No submission (on paper or via e-mail)
    - No automatic group assignment
- Optimus: Automatic group assignment
- ExClaim & RTE: For submission & testing

Three separate systems!

- Users management only in STATS
- Linking data via student id

History (2)

- Integrate all features in one system
- Optimize database layout
- Use code-generation for type-safe database access
- Simplify building the system
- New Frontend in development

Your Task (1): Data Generation

- Initialize Database with dummy data
- Facilitates development
- For demonotration purposes
- (Cannot use real data!)

Your Task (2): Data Export

- Retrieve all data for a single course (for offline evaluation)
- Retrieve all data for a single user (to fulfill GDPR requests)
- Retrieve essential data for a single course (Overall homework points per student, exam assessment, but no uploads or single sheet points)
- Format: JSON or XML

Your Task (3): Data Deletion

- Delete entire course
- from database (recursively all dependent data) and stored files (homework uploads)
- Required to delete data after retention period
- To not lose exam admissions, archived export of essential data is important

Technologies to Learn / Know

- Git
- Backend
    - Java
    - Spring Boot
    - jOOQ (Code Generator for type-safe database access)
- Frontend
    - TypeScript / Javascript
    - Vue
    - HTML, CSS

Search tool for Agda

Supervisor: Alexander Dinges

Your task

- CLI search tool for an Agda repository
- Find function signatures/propositions/types together with their names given a part of the type
- Reasonably fast

Minimum requirements for getting your credit points

- Given a signature $t_1 \rightarrow t_2 \rightarrow \ldots t_n$ find all (complete) signatures (in a given Agda code base) that contain $t_1, \ldots t_{n-1}$ as parameter type and t_n as return type.
  - $t_1, \ldots, t_n$ match up to variable renaming (consistently) and number representation
  - I.e. find $1 \prec n \rightarrow 0 \prec n/2$ when given $(\mathsf{succ}\ \mathsf{zero}) \prec a \rightarrow \mathsf{zero} \prec a/2$
- Don't find irrelevant stuff.
- Reasonable ranking and speed.

Minimum requirements for getting your credit points

Moreover, choose at least 2 of the following points:

- Make the CLI user-friendly: Agda input method, colors, search history, ...
- Allow more flexible search strings: Partial matching, underscore patterns, ...
- Good ranking.
- Allow more unification
- Your own idea?

Technologies to learn/know

- Git
- A tiny bit of Agda
- You can use (almost) any programming language you like

Developing a web app using Haskell

Supervisor: Michael Youssef

Why Haskell?

- Haskell is a pure language
- No unintended side effects
- ADTs
- Modularity
- Laziness
- Referential transparency

Why Haskell?

- Haskell is a pure language
- No unintended side effects
- ADTs
- Modularity
- Laziness
- Referential transparency
- No null-pointer exceptions
- No IO exceptions
- Elegant approach towards handling IO
- No unhandled cases

Challenges/What you will learn?

- Type classes
- Practical usages of functors, applicatives and monads
- Monad transformers
- Type families
- Template Haskell
- Debugging with lazy evaluation

Challenges/What you will learn?

- Type classes
- Practical usages of functors, applicatives and monads
- Monad transformers
- Type families
- Template Haskell
- Debugging with lazy evaluation

- Network IO
- Database storage
- Caching
- Writing a fully fledged application in Haskell

Why should you care?

- ECTS....
- Learn some practical applications of the stuff you learned in FP
- Knowledge you gain is transferable to other programming language paradigms

LaTeX-formatted Execution Traces of Algorithms on Automata & Grammars

Supervisor: Cass Alexandru

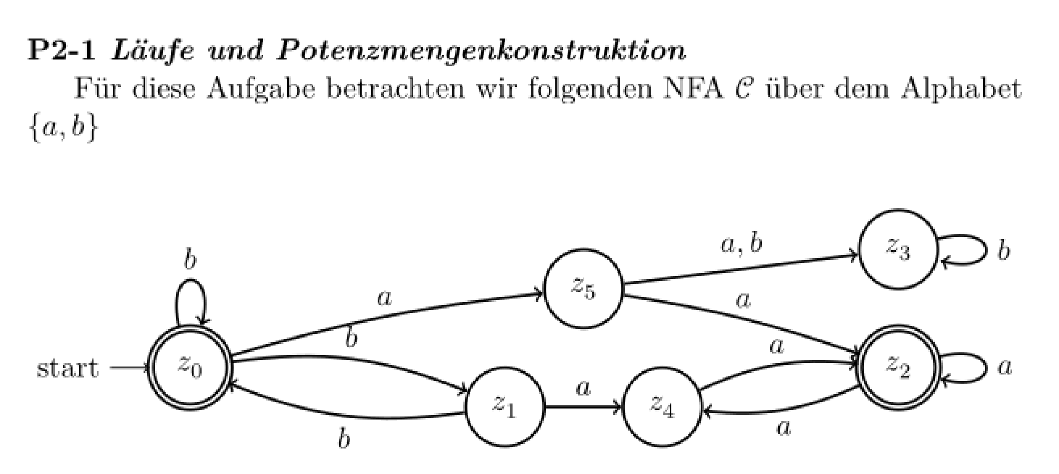
LaTeX-formatted Execution Traces of Algorithms on Automata & Grammars

Problem Statement

- Lecture “Formal Languages and Computability”
- Students learn to execute several algorithms on automata & grammars by hand
- I (the client) want a software tool that:
    - Given an input object (an automaton or grammar)
    - Executes one of a number of algorithms on it and outputs its execution trace (showing its steps) in the format used in the lecture
- This tool would allow me to more quickly iterate exercise ideas & easily and confidently generate correct reference solutions without time-consuming and error-prone manual calculation

Algorithms

- Automata:
  - Minimization
  - Determinization (Powerset/Rabin-Scott Construction)
  - Product Automaton
  - Execution trace ((nondeterministic) stack automaton)
  - Execution trace (Turing machine)
- Grammars:
  - CNF (Chomsky Normal Form) algorithm for context-free grammar
  - CYK Algorithm for bottom-up parsing of words in a cf grammar
    - Derivation of word from starting symbol in case of language membership



P2-1 ***Läufe und Potenzmengenkonstruktion***

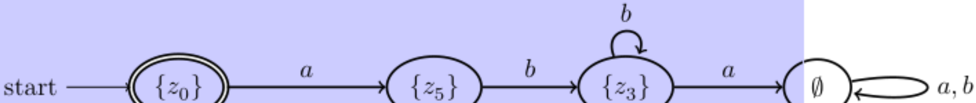
Für diese Aufgabe betrachten wir folgenden NFA $\mathcal{C}$ über dem Alphabet $\{a, b\}$

LÖSUNGSVORSCHLAG:

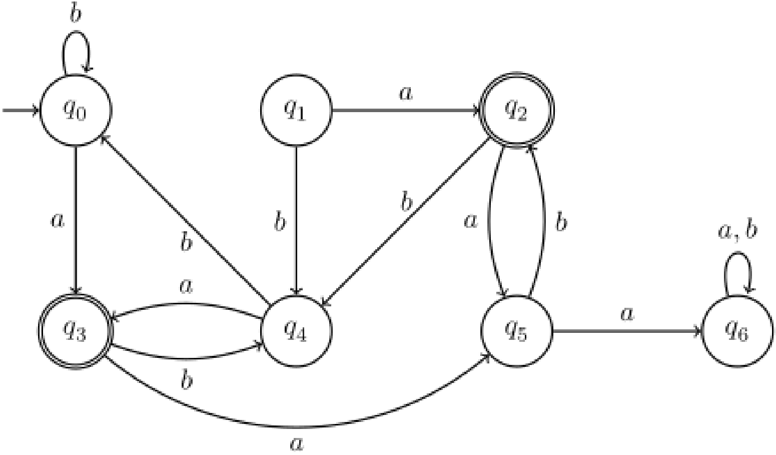
Potenzmengenkonstruktion, wobei nur die erreichbaren Zustände betrachtet werden:

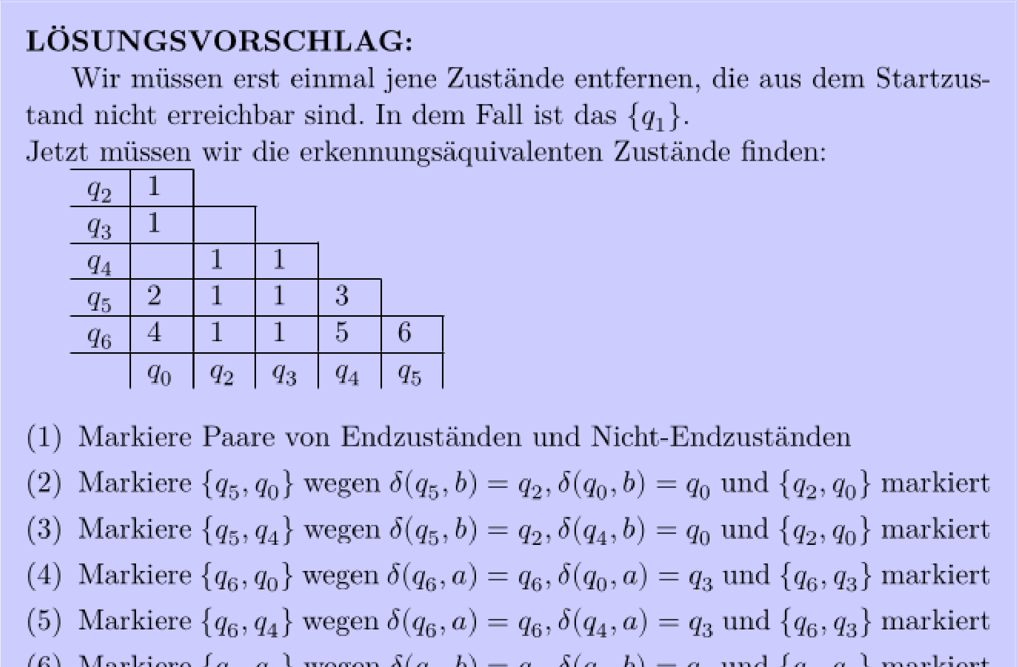
Zustand/Nachfolge bei	a	b
$\{z_0\}$	$\{z_5\}$	$\{z_0, z_1\}$
$\{z_5\}$	$\{z_2, z_3\}$	$\{z_3\}$
$\{z_0, z_1\}$	$\{z_5, z_4\}$	$\{z_0, z_1\}$
$\{z_2, z_3\}$	$\{z_2, z_4\}$	$\{z_3\}$
$\{z_3\}$	$\emptyset$	$\{z_3\}$
$\{z_5, z_4\}$	$\{z_2, z_3\}$	$\{z_3\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\{z_2, z_4\}$	$\{z_2, z_4\}$	$\emptyset$

Startzustand $\{z_0\}$ und Endzustände: $\{z_0\}, \{z_0, z_1\}, \{z_2, z_3\}, \{z_2, z_4\}$



P6-1 ***Automaten minimieren*** Minimieren Sie folgenden DFA $\mathcal{A}$ nach dem Verfahren aus der Vorlesung. Geben Sie den minimierten Automaten an.





LÖSUNGSVORSCHLAG:

Wir müssen erst einmal jene Zustände entfernen, die aus dem Startzustand nicht erreichbar sind. In dem Fall ist das $\{q_1\}$.
Jetzt müssen wir die erkennungsäquivalenten Zustände finden:

q_2	1				
q_3	1				
q_4		1	1		
q_5	2	1	1	3	
q_6	4	1	1	5	6
	q_0	q_2	q_3	q_4	q_5

(1) Markiere Paare von Endzuständen und Nicht-Endzuständen

(2) Markiere $\{q_5, q_0\}$ wegen $\delta(q_5, b) = q_2, \delta(q_0, b) = q_0$ und $\{q_2, q_0\}$ markiert

(3) Markiere $\{q_5, q_4\}$ wegen $\delta(q_5, b) = q_2, \delta(q_4, b) = q_0$ und $\{q_2, q_0\}$ markiert

(4) Markiere $\{q_6, q_0\}$ wegen $\delta(q_6, a) = q_6, \delta(q_0, a) = q_3$ und $\{q_6, q_3\}$ markiert

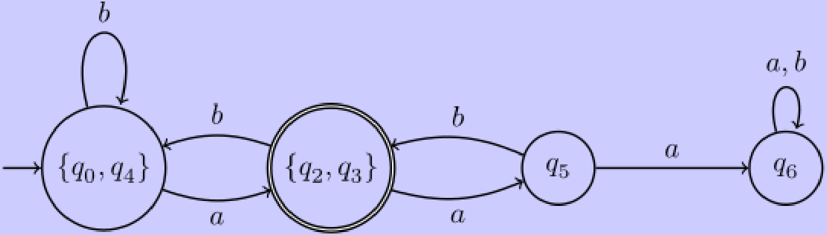
(5) Markiere $\{q_6, q_4\}$ wegen $\delta(q_6, a) = q_6, \delta(q_4, a) = q_3$ und $\{q_6, q_3\}$ markiert

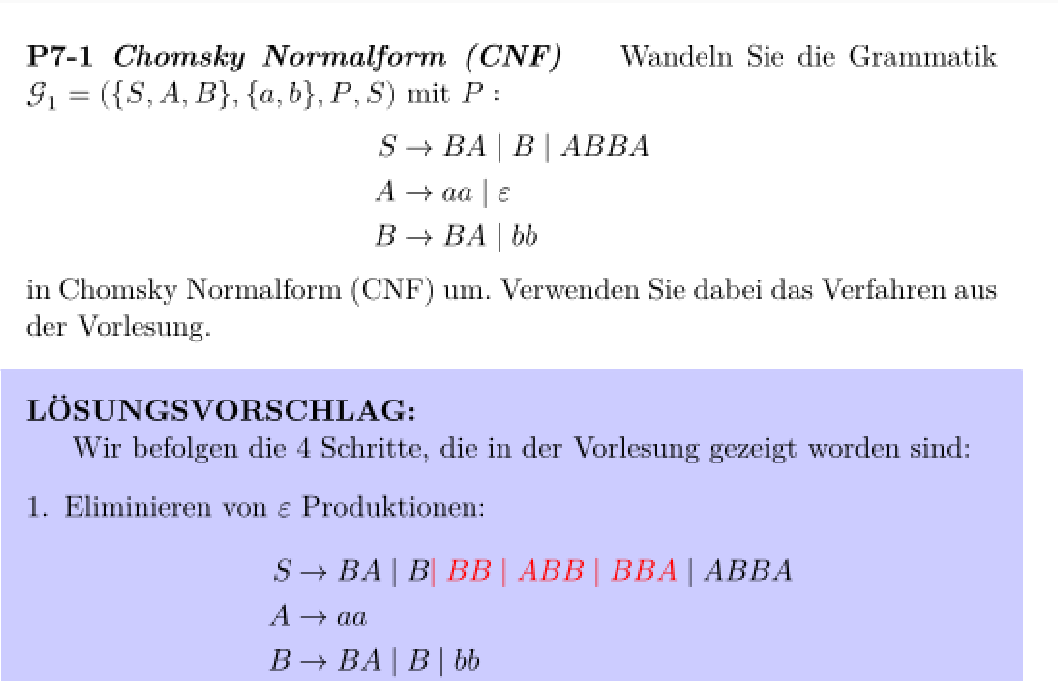
(6) Markiere $\{q, q\}$ wegen $\delta(q, b) = q, \delta(q, b) = q$ und $\{q, q\}$ markiert

Wir können also die folgenden Zustände verschmelzen:

- q_0 mit q_4
- q_2 mit q_3

Und erhalten folgenden Minimalautomat:





P7-1 ***Chomsky Normalform (CNF)*** Wandeln Sie die Grammatik $\mathcal{G}_1 = (\{S, A, B\}, \{a, b\}, P, S)$ mit P :

$$\begin{aligned} S &\to BA \mid B \mid ABBA \\ A &\to aa \mid \varepsilon \\ B &\to BA \mid bb \end{aligned}$$

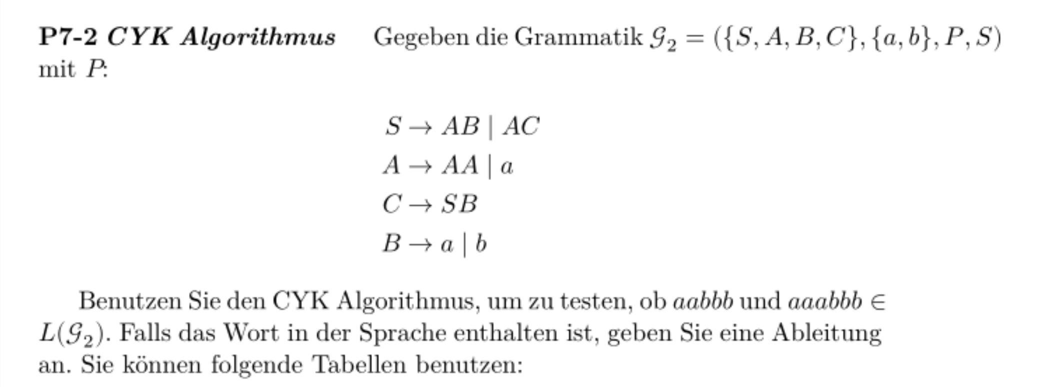
in Chomsky Normalform (CNF) um. Verwenden Sie dabei das Verfahren aus der Vorlesung.

LÖSUNGSVORSCHLAG:

Wir befolgen die 4 Schritte, die in der Vorlesung gezeigt worden sind:

1. Eliminieren von ε Produktionen:

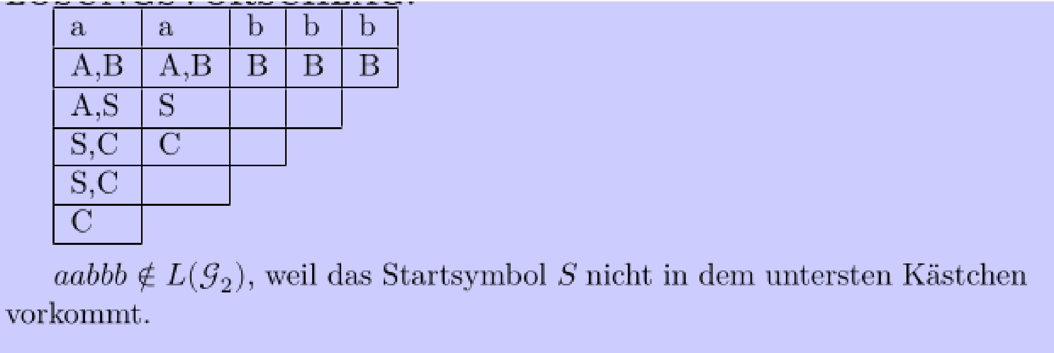
$$\begin{aligned} S &\to BA \mid B \mid BB \mid ABB \mid BBA \mid ABBA \\ A &\to aa \\ B &\to BA \mid B \mid bb \end{aligned}$$



P7-2 *CYK Algorithmus* Gegeben die Grammatik $\mathcal{G}_2 = (\{S, A, B, C\}, \{a, b\}, P, S)$ mit P:

$$\begin{aligned} S &\to AB \mid AC \\ A &\to AA \mid a \\ C &\to SB \\ B &\to a \mid b \end{aligned}$$

Benutzen Sie den CYK Algorithmus, um zu testen, ob $aabbb$ und $aaabbb \in L(\mathcal{G}_2)$. Falls das Wort in der Sprache enthalten ist, geben Sie eine Ableitung an. Sie können folgende Tabellen benutzen:



a	a	b	b	b
A,B	A,B	B	B	B
A,S	S			
S,C	C			
S,C				
C				

$aabbb \notin L(\mathcal{G}_2)$, weil das Startsymbol S nicht in dem untersten Kästchen vorkommt.

LÖSUNGSVORSCHLAG:

PDA

$\mathcal{P}_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{\#, A, B\}, q_0, \Delta, \#)$ mit Δ:

$$
\begin{aligned}
(q_0, a, \#) &\rightarrow (q_0, A\#) \\
(q_0, a, A) &\rightarrow (q_0, AA) \\
(q_0, b, A) &\rightarrow (q_1, BA) \\
(q_1, b, B) &\rightarrow (q_1, BB) \\
(q_1, c, B) &\rightarrow (q_2, \varepsilon) \\
(q_2, c, B) &\rightarrow (q_2, \varepsilon) \\
(q_2, d, A) &\rightarrow (q_3, \varepsilon) \\
(q_3, d, A) &\rightarrow (q_3, \varepsilon) \\
(q_3, \varepsilon, \#) &\rightarrow (q_3, \varepsilon)
\end{aligned}
$$

Wörter in L_1:

LaTeX-formatted Execution Traces of Algorithms on Automata & Grammars

Execution

Required Skills

- At least one member should have experience with the following:
    - B2 level German (reference solns are generated in German, though you won't need to write much text yourself, I provide templates)
    - LaTeX (again, I mostly provide templates and will also provide guidance)
    - The algorithms in question from the "Formal Languages" lecture

Development Process

- Regular meetings with me to keep shared understanding of requirements synched
- There will be milestones for deliverables with intermediate functionality
- Development should use Gitlab Issues, Issue Boards and, ideally, CI/CD, with provisions made for shared, replicable build and dev environments across the team and CI

Roadmap

- 23.05: DFA Minimization
- 30.05: CNF
- 06.06: CYK
- 13.06: PDA trace
- 20.06: TM trace
- 27.06: NFA Determinization
- 04.07: Product Automaton

Divide up responsibilities & pipeline tasks!